Se citeste un numar natural n. Afisati permutarile multimii 1,2,3...n in care elementele pare sunt puncte fixe (se afla pe pozitie egale cu valoarea lor).   
Exemplu:   
n=5   
permutarile care respecta conditia sunt:   
1 2 3 4 5   
1 2 5 4 3   
3 2 1 4 5   
3 2 5 4 1   
5 2 1 4 3   
5 2 3 4 1   
(2 si 4 sunt puncte fixe)

#include <fstream>

using namespace std;

ifstream fin("date.in");

ofstream fout("date.out");

int n, sol[40], p[40];

void afis()

{

for(int i=1;i<=n;i++)

if(i%2==0) fout<<i<<" ";

else fout<<sol[i]<<" ";

fout<<endl;

}

void back(int k)

{

for(int i=1;i<=n;i=i+2)

if(p[i]==0)

{

sol[k]=i;

p[i]=1;

if(n%2==0 && k==n-1 || n%2==1 && k==n) afis();

else back(k+2);

p[i]=0;

}

}

int main()

{

fin>>n;

back(1);

fin.close();

fout.close();

}

Se citeste un numar natural n. Afisati permutarile multimii 1,2,3...n in care pana la jumatate elementele sunt in ordine descrescatoare, iar de la jumatate pana la final in ordine crescatoare.   
Exemplu:   
n=5   
permutarile care respecta conditia sunt:   
3 2 1 4 5   
4 2 1 3 5   
4 3 1 2 5   
5 2 1 3 4   
5 3 1 2 4   
5 4 1 2 3

#include <fstream>

using namespace std;

ifstream fin("date.in");

ofstream fout("date.out");

int n, sol[20], p[20];

void afis()

{

for(int i=1;i<=n;i++)

fout<<sol[i]<<" ";

fout<<endl;

}

int valid(int k)

{

if(k>1)

if(k<=(n+1)/2) { if(sol[k]>sol[k-1]) return 0; }

else if(sol[k]<sol[k-1]) return 0;

return 1;

}

void back(int k)

{

for(int i=1;i<=n;i++)

if(p[i]==0)

{

sol[k]=i;

p[i]=1;

if(valid(k)) if(k==n) afis();

else back(k+1);

p[i]=0;

}

}

int main()

{

fin>>n;

back(1);

fin.close();

fout.close();

}

|  |  |
| --- | --- |
| |  | | --- | | Se citesc doua numere naturale n si k. Generati si afisati toate toate numerele naturale formate din n cifre care contin exact k cifre de 1. | |

#include <iostream>

using namespace std;

int x[50],n,k;

void afis()

{

for(int i=1;i<=n;i++)

cout<<x[i];

cout<<endl;

}

int valid(int pas)

{

int c=0;

for(int i=1;i<=pas;i++)

if(x[i]==1) c++;

if(c>k) return 0;

if(pas==n && c!=k) return 0;

if(x[1]==0) return 0;

return 1;

}

void back(int pas)

{

for(int i=0;i<=9;i++)

{

x[pas]=i;

if(valid(pas)) if(pas==n) afis();

else back(pas+1);

}

}

int main()

{

cin>>n>>k;

back(1);

return 0;

}

Un student are de dat n examene numerotate de la 1 la n intr-o sesiune formata din m zile (m este cel putin de 2 ori mai mare decat n).   
Afisati toate modurile in care isi poate programa studentul examenele astfel incat sa nu dea 2 examene in zile consecutive si sa dea examenele in ordine de la 1 la n.   
Ex:   
n=3   
m=6   
Solutii:   
010203   
100203   
102003   
102030   
(0 codifica zilele libere)

#include<fstream> using namespace std; ifstream fin("date.in");ofstream fout("date.out");

int p[100];

void afis(int x[], int n, int m) {

for(int i=1;i<=m;i++) fout<<x[i];

fout<<endl;}

int bun(int x[], int k, int n, int m) {

if(k>1)

{

if(x[k-1]\*x[k]!=0) return 0;//2 examene alaturate

if(x[k]>1)//vreau sa pun examen

{ //caut examen cu 1 mai mic

for(int i=1;i<k;i++) if(x[i]==x[k]-1) return 1;

return 0;

} }

if(x[1]>1) return 0;//nu pot incepe cu >1

if(x[k]==0)//vreau sa pun zi libera

{

int s=0;

for(int i=1;i<=k;i++) if(x[i]==0) s++;

if(s>m-n) return 0;//nu pot fi mai multi de 0 de m-n

}

return 1; }

void back(int x[], int n, int m, int k) {

int i;

for(i=0;i<=n;i++)

if(!p[i]||i==0) //examenele nu se repeta

{ x[k]=i;

p[i]=1;

if(bun(x,k,n,m)) if (k==m) afis(x,n,m); else back(x,n,m,k+1);

p[i]=0; } }

int main() {

int x[100],n,m; fin>>n>>m;

back(x,n,m,1);

fin.close();

fout.close();

return 0;

}

Se citeste un numar natural n. Afisati toate numerele formate din n cifre in care oricare doua cifre alaturate au paritate diferita.   
Ex. Pentru n=4 se vor obtine:   
1010   
1012   
1014   
...   
2101   
2103   
...   
9898

#include<fstream>

using namespace std;

ifstream fin("date.in");

ofstream fout("date.out");

void afis(int x[], int n)

{

for(int i=1;i<=n;i++) fout<<x[i];

fout<<endl;

}

int bun(int x[], int k)

{

if(x[1]==0) return 0;

if(k>1) if(x[k-1]%2==x[k]%2) return 0;

return 1;

}

void back(int x[], int n, int k)

{

int i;

for(i=0;i<=9;i++)

{

x[k]=i;

if(bun(x,k)) if (k==n) afis(x,k);

else back(x,n,k+1);

}

}

int main()

{

int x[100],n;

fin>>n;

back(x,n,1);

fin.close();

fout.close();

return 0;

Se citeste un numar natural n. Sa se afiseze toate modurile in care poate fi descompus ca suma de numere naturale impare distincte.

#include<iostream>

using namespace std;

int n, x[100];

void afis(int k)

{

for(int i=1;i<=k;i++) cout<<x[i]<<" ";

cout<<endl;

}

void back(int k, int sp)

{

if(sp==n) afis(k-1);

else

for(int i=1;i<=n;i=i+2)

{

x[k]=i;

if(sp+x[k]<=n && x[k]>x[k-1]) back(k+1,sp+x[k]);

}

}

int main()

{

cin>>n;

back(1,0);

return 0;

}

Despre fiecare dintre cei n elevi dintr-o clasa se citesc urmatoarele date:   
- numele (sir de caractere de lungime maxima 20)   
- numele (sir de caractere de lungime maxima 20)   
- media notelor in oral la informatica   
- nota in teza la informatica   
Sa se afiseze ordonati alfabetic elevii corigenti la informatica. Pentru fiecare elev corigent se vor afisa: numele, prenumele si media la informatica.

|  |  |
| --- | --- |
| |  | | --- | | #include<fstream.h>  #include<string.h>  struct elev {char nume[30];  char pren[30];  float mo;  int teza;  float media;  };  ifstream f("elev.in");  elev c[100];  void citire(elev &e)  {f>>e.nume;  f>>e.pren;  f>>e.mo;  f>>e.teza;  e.media=(e.mo\*3+e.teza)/4;  }  void afisare(elev e)  {cout<<e.nume<<" "<<e.pren<<" "<<e.media;  cout<<endl;  }  void ordonare(int n,elev c[100])  {int i,gata;  elev aux;  do {gata=1;  for (i=1;i<n;i++)  if (strcmp(c[i].nume,c[i+1].nume)>0)  {aux=c[i];  c[i]=c[i+1];  c[i+1]=aux;  gata=0;  }  }while (!gata);  }  int main ()  { int n,i;  f>>n;  for (i=1;i<=n;i++) citire(c[i]);  ordonare (n,c);  for (i=1;i<=n;i++)  if (c[i].media<4.5)  afisare(c[i]);  return 0;  } | |

PBINFO

**#123 Permutari**

Se citeşte un număr natural nenul *n*. Să se afişeze, în ordine lexicografică, permutările mulţimii *{1,2,..,n}*.

#include <fstream> using namespace std;

ifstream fin("permutari.in"); ofstream fout("permutari.out");

int n,sol[11];

void afisare()

{

for(int i=1; i<=n; i++)

fout<<sol[i]<<' ';

fout<<endl;

}

int valid(int k)

{

for(int i=1; i<=k-1; i++)

if(sol[k]==sol[i])

return 0;

return 1;

}

void BKT(int k)

{

if(k==n+1)

afisare();

else

for(int i=1; i<=n; i++)

{

sol[k]=i;

if(valid(k))

BKT(k+1);

}

}

int main()

{

fin>>n;

BKT(1);

}

**124** Permutari1

Se citeşte un număr natural nenul *n*. Să se afişeze, în ordine invers lexicografică, permutările mulţimii*{1,2,..,n}*.

#include <fstream>

using namespace std;

ifstream fin("permutari1.in");

ofstream fout("permutari1.out");

int n,sol[11];

void afisare()

{

for(int i=1; i<=n; i++)

fout<<sol[i]<<' ';

fout<<endl;

}

int valid(int k)

{

for(int i=1; i<=k-1; i++)

if(sol[k]==sol[i])

return 0;

return 1;

}

void BKT(int k)

{

if(k==n+1)

afisare();

else

for(int i=n; i>=1; i--)

{

sol[k]=i;

if(valid(k))

BKT(k+1);

}

}

int main()

{

fin>>n;

BKT(1);

}

**#1287** [**CifreGen**](http://www.pbinfo.ro/?pagina=probleme&id=1287)

Se dau două cifre *a b* și un număr *n*. Să se genereze toate numerele cu exact *n* cifre cuprinse între *a* și *b*.

|  |  |
| --- | --- |
| #include <iostream>  #include <fstream>  #include <algorithm>  #define nmax 11  using namespace std;  int x[nmax],n,a,b;  void perm(int k)  {  if(k>n)  {  for(int j=1; j<=n; j++)  cout<<x[j];  cout<<"\n";  }  else  for(int i=a; i<=b; i++)  {  x[k]=i;  perm(k+1);  }  }  int main()  {  //freopen("permutari.in","rt",stdin);  //freopen("permutari.out","wt",stdout);  cin>>a>>b>>n;  perm(1);  return 0;  } | 3 5 3  **Ieșire**  333  334  335  343  344  345  353  354  355  433  434  435  443  444  445  453  454  455  533  534  535  543  544  545  553  554  555 |

### #320 [PartitiiNumar](http://www.pbinfo.ro/?pagina=probleme&id=320)

Se dă un număr natural *n*. Determinaţi, în ordine lexicografică, toate modalităţile de a-l scrie pe *n* ca sumă de numere naturale.

|  |  |
| --- | --- |
| #include <iostream>  #include <fstream>  #include <stdlib.h>  using namespace std;  ifstream fin ("partitiinumar.in");  ofstream fout ("partitiinumar.out");  int x[41], n;  void part(int k, int s)  {  int i;  if(s==0)  {for(i=1; i<k; i++)  fout<<x[i]<<" ";  fout<<endl;  }  else  for(i=x[k-1]; i<=s; i++)  {  x[k]=i;  part(k+1, s-i);  }  }  int main()  {  fin>>n;  x[0]=1;  part(1,n);  } | *partitiinumar.in*  5  *partitiinumar.out*  1 1 1 1 1  1 1 1 2  1 1 3  1 2 2  1 4  2 3  5 |

# #319 suma35

Se dă un număr natural nenul *n*. Să se determine toate modalităţile distincte de descompunere a numărului *n* în sumă de *3* şi *5*.

|  |  |
| --- | --- |
| #include <iostream>  #include <fstream>  #include <algorithm>  #define nmax 1005  using namespace std;  int x[nmax],n,s;  void partitiinr(int k)  {  if(s==n)  {  for(int i=1; i<k; i++)  cout<<x[i]<<' ';  cout<<'\n';  }  else if(k<=n && s<n)  {  if(x[k-1]!=5)  {  x[k]=3;  s+=x[k];  if(s<=n)  partitiinr(k+1);  s-=x[k];  }  x[k]=5;  s+=x[k];  if(s<=n)  partitiinr(k+1);  s-=x[k];  }  }  int main()  {  freopen("suma35.in","rt",stdin);  freopen("suma35.out","wt",stdout);  cin>>n;  partitiinr(1);  return 0;  } | *suma35.in*  40  *suma35.out*  3 3 3 3 3 3 3 3 3 3 5 5  3 3 3 3 3 5 5 5 5 5  5 5 5 5 5 5 5 5 |